JOURNAL OF SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE
*J. Softw. Maint: Res. Pract.* **11**, 285–291 (1999)

**Book Review**

# Software Process Improvement: Practical Guidelines for Business Success

**Addison Wesley Longman, UK, 1997, £27.95, 447 pp.**

**Author**: SAMI ZAHRAN*
*International Business Machines Corporation, London, UK.*

**SUMMARY**

**This 1997 book advocated the use of disciplined processes in doing software work, and of continued improvement in those processes. The author explains that in achieving software process improvement (SPI), the primary emphasis has to be on changing the behaviour of people—effectively a maintenance process. This book concentrates not on tools or technology, but on working together toward goals. While the book draws upon both European and American experience, the Capability Maturity Model (CMM) of the Software Engineering Institute (SEI) receives the most attention. This book, in spite of the word 'business' in its title, is as applicable to governmental, non-profit, research, commercial, and industrial organizations as it is to business. The real focus of this book is on discipline in software work processes wherever they may be found, a condition that appears to be applicable in software maintenance processes, although software maintenance processes are not this book's nominal domain. Copyright © 1999 John Wiley & Sons, Ltd.**

KEY WORDS: disciplined work processes; improvement of processes; process models; process standards; capability maturity model; group work processes

## 1. OVERVIEW OF BOOK

This book is part of the Software Engineering Institute (SEI) Series in Software Engineering from the UK office of its publisher, Addison Wesley Longman. As would be assumed and as the author, Dr. Sami Zahran of IBM in the UK confirms, this book's contents have been strongly influenced by the SEI and its Capability Maturity Model (CMM). Although the word 'business' is in the title, the book's applicability is far broader and includes all kinds of organizations that develop or maintain software for themselves or for their customers.

This clearly written and well-produced book is effectively Sami Zahran's take on using disciplined processes in software work. For that reason, code cowboys and cowgirls and their managers have but faint praise for this book. The author, Sami Zahran, has not limited himself

*Correspondence to: Dr Sami Zahran, c/o Helen Clatworthy, Higher Education Division, Addison Wesley Longman, Edinburgh Gate, Harlow, Essex CM20 2JE, UK. Email: helen.clatworthy@awl.co.uk.

CCC 1040–550X/99/040285–07$17.50

to the SEI model with its CMM. He also has drawn upon his own three decades of experiences and observations, and upon other sources, including:

- the ISO/IEC 15504 Draft Standard for Software Process Assessment, as put forward by SPICE (Software Process Improvement and Capability dEtermination);
- the BOOTSTRAP model that arose from the European Community project ESPRIT (Project # 5441);
- the ISO 9000 and 9001 quality standards;
- the ISO/IEC 12207 standard life cycle processes;
- the USA DoD (Department of Defense) MIL–STD–498 military standard on software development and documentation;
- the Canadian Trillium model, a model based on the SEI CMM; and
- the German V-Model, a software life cycle process model.

The author's aim in writing this book was to offer a pragmatic approach for the effective implementation of software process improvement (SPI) in a continuously improving but disciplined software process environment. The author has put his focus on the roles of people and frameworks for making changes in software processes. His key point of advocacy is having a work culture integrated via disciplined software processes.

To argue his case, the author offers twenty chapters, a preface, two forewords, a glossary, and a list of references, along with the usual table of contents and index. The preface is ten pages long and includes a 'how to use this book' and extensive acknowledgements. The first foreword, two pages long, is by Watts S. Humphrey. The second foreword, five pages long, is by Mark C. Paulk. Both have been closely associated with the SEI and its CMM. The 110 references take up six pages, with Humphrey and Paulk accounting for over sixteen per cent of them. The 262 glossary items take up 26 pages and summarize the way the author has used some terms in this book. The table of contents at six pages lists the major headings within the chapters. The five-page index has only about 425 entries and is so wimpy it is frustrating to use.

The author has grouped the twenty chapters into five parts, as summarized in Table 1. Each of the five parts is described briefly later in this review. The author suggests five possible orders for reading the five sections. Figure 1 illustrates these. The sequence of parts 1, 3 and 5 (34% of the book) appears to be what the author advocates as the main route to SPI, with parts 2 and 4 (66% of the book) providing supporting material. This reviewer wonders at this incongruence between space devoted and stated importance.

## 2. OVERALL REVIEW

About this book, a key question is 'What is its relevance for software maintenance?' Table 1 indicates no explicit and extensive coverage of software maintenance. All of the terms 'maintenance', 'evolution', and 'support' in the sense of maintenance are missing from the index. A three-line entry in the glossary for 'maintenance' is present, but 'evolution' and 'support' are absent. However, in the glossary on p. 419 under 'development' the author briefly states that he regards making enhancements as development.

As noted in the prior section, the author of this book has been strongly influenced by the SEI and its CMM. The CMM, as has been pointed out elsewhere as for example by Masengale and

 

Table 1. This book groups its twenty chapters into five clusters it terms ‘parts’

| Part | Page length | Part names with the chapter numbers and titles |
|---|---|---|
| 1 | 64 | PROCESS THINKING |
| | | 1—Process thinking |
| | | 2—Process discipline |
| | | 3—Effective process environment |
| | | 4—Process maturity: the second wave of the software industry |
| 2 | 116 | A FRAMEWORK FOR SOFTWARE PROCESS IMPROVEMENT |
| | | 5—A framework for software process improvement |
| | | 6—Software process infrastructure |
| | | 7—Process improvement roadmaps |
| | | 8—Fundamentals of software process assessment |
| | | 9—Software process improvement action plan |
| 3 | 52 | MAKING SOFTWARE PROCESS IMPROVEMENT HAPPEN |
| | | 10—Launching software process improvement |
| | | 11—Implementing and institutionalizing software process improvement |
| | | 12—Measuring the benefits of software process improvement |
| 4 | 154 | CURRENT MODELS AND STANDARDS FOR SOFTWARE PROCESS IMPROVEMENT |
| | | 13—The capability maturity model (CMM) |
| | | 14—CMM-based software process assessment |
| | | 15—ISO/IEC 15504 draft standard for software process assessment |
| | | 16—ISO/IEC 15504 draft guide to conducting assessment |
| | | 17—BOOTSTRAP software process assessment |
| | | 18—Other initiatives |
| 5 | 24 | BUSINESS BENEFITS OF SOFTWARE PROCESS IMPROVEMENT |
| | | 19—The evidence: business benefits of software process improvement |
| | | 20—Epilogue: future of software process improvement |

Schrank (1997), gives short shift to maintenance, even though far more is spent on maintenance than on development by the SEI’s primary sponsor, the USA’s DoD. Furthermore, as has also been pointed out elsewhere as for example by Chapin (1988), trying to manage software maintenance like software development raises costs and decreases benefits. Also, since the SDLC (software development life cycle) is not the same as the SMLC (software maintainance life cycle), the appropriateness of using the same processes in the same ways for both is open to question.

Just because someone’s situation is different does not mean that we cannot learn something anyway. For those of us concerned with software maintenance, this book does give us grounds for considering the possible contributions in maintenance of disciplined processes, of applying processes in a disciplined manner, and of SPI. While the choice of some specific development processes may not be applicable in some software maintenance circumstances, is the practice of discipline in processes and in SPI applicable in software maintenance? The author of this book does not address these maintenance matters directly (software maintenance processes are not the book’s domain), but argues cogently for consequences that look attractive or have potential for possible transfer to software maintenance. It is that argument that pervades this book and understanding it is the primary value of and reason for reading it, in the opinion of this reviewer.

 

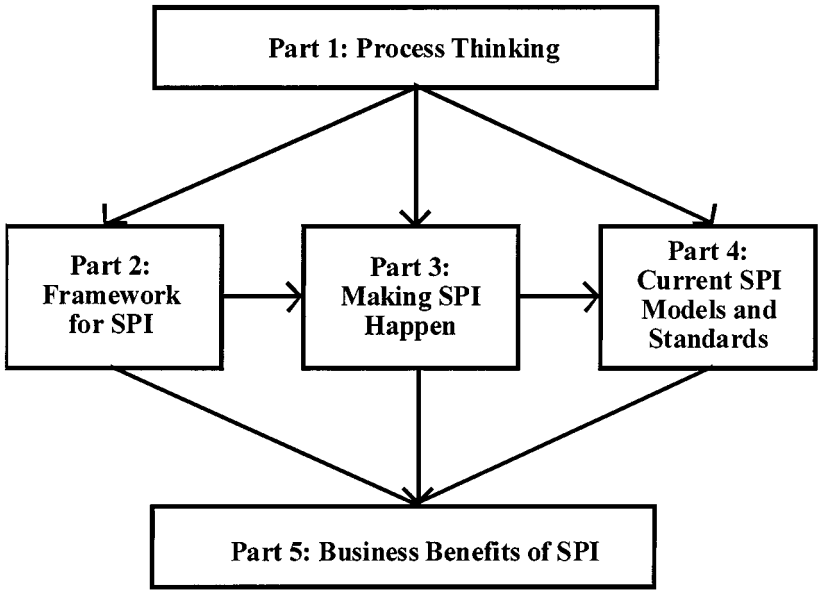


*Figure 1. The author's guide for using this book on SPI (software process improvement) offers the reader six routes through the book*

## 3. THRUST OF THE FIVE PARTS

### 3.1. Process thinking

The first part of the book is about thinking in terms of processes. By process, the author means human group behaviour and attitudes that implement a set of steps in a specific sequence toward a specified goal. The author uses the term 'process' for ways of getting work done when a group of people are involved, where a programmer would use the term 'algorithm' for ways of getting work done by a computer—but with one major difference. That difference is that people frequently and repeated change the process in attempts to improve it, but programmers rarely change the algorithm, and when they do, it is also usually in an attempt to improve it.

Three key aspects of process thinking are:

- defining the process, in the equivalent of a written accessible form;
- training the group members in doing their parts of the process; and
- enforcing and monitoring the performance of the process.

Having the group perform in accord with a process in doing software work gives stability in the organization and reduces stress in the organization, the author observes.

The discipline arises from each member of the group knowing what performance is expected of and from each member of the group as they follow the ordered patterns of behaviour called for in the process. That discipline gives a dependable level of quality in the product produced by the group,

 

if the process is effective and supported by the organization's infrastructure. For the process to be effective, it must:

- have management support, from top to bottom in the organization;
- be part of the organization's culture;
- have goals aligned with the organizations's goals;
- be responsive to feedback from the process users and the external environment to confirm that benefits are realized; and
- have some member of management committed to be the champion or 'owner' of the process.

The continued improvement in software processes leads to increasing maturity in the processes and improved consistency in product quality, the author observes.

## 3.2. Framework for SPI

To get continued improvement in the processes—i.e., to maintain the processes—requires four environmental elements to be in place:

- the software process infrastructure;
- the software process improvement models and standards;
- the software process assessment method; and
- the software process improvement plan.

The infrastructure should include more than software tools. It should include technical support for them and their use, and technical support on the use of both software engineering methods and techniques, and on the steps in the processes. This requires a dedicated staff and payment of the associated expenses. A wealth of process improvement models and standards exist relevant to software quality and ways to improve it, leading up to the SEI's CMM in the author's view.

A key step (a 'launch pad' the author terms it) in improving software processes is to assess the processes, but no assessment is needed if no change is to be made to the processes. The phases the author lists in a generic assessment are pre-planning, planning, fact gathering, analyzing to get the assessment, reporting the assessment, and making a post-assessment if the assessment was acted upon. If the assessment is acted upon, the assessment must be converted into an improvement project, with roles and responsibilities clearly specified and assigned. When the software personnel using an existing process may be the ones who act to improve it based upon the assessment, sometimes a 'PIT' (process improvement team) may be involved.

## 3.3. Making SPI happen

The main steps in making software process improvement happen are:

- launching;
- implementing;
- institutionalizing;
- measuring; and
- improving.

 

Launching the improvement in a software process calls for a selling effort to gather the support and commitment of the bulk of the affected personnel and of all affected senior management personnel. Promoting a link between the organization's vision and strategy and the proposed software process improvement and its organizational context is an aid in selling the improvement project. To mitigate fears, SPI projects need well defined boundaries and well understood models. The author lists ten 'critical success factors' for launching a successful SPI project.

Since an SPI project is a project for changing the on-the-job behaviour of software personnel and their supervisors, human considerations are vital in achieving project objectives during implementation. Resistance to change is normal and must be overcome. Institutionalizing the changes can help make change palatable. In practice, SPI is a combination of two kinds of maintenance: maintenance of the ways software personnel apply their skills, and maintenance of the organization. For those reasons and to sustain them, any SPI project should measure the before and after situation, interpret the differences, and communicate them, even though resistance to measurement can be anticipated from some affected persons. To reduce resistance, what is measured, how it is measured, and how differences are interpreted need to be carefully thought out, sold, and carried out.

## 3.4. Current models and standards for SPI

As noted earlier, the author has placed a lot of emphasis on what he terms the 'models and standards' for the guidance they can offer on SPI. The author makes them one of his big five parts and in terms of pages space, gives them the most attention of the five (see Table 1). In this fourth of five parts, the author gives the most attention to the SEI work and its CMM. But the author does not neglect summarizing the ISO/IEC 15504, the ISO 9000 and 9001, the BOOTSTRAP, the MIL–STF–498, the V-Model, the Trillium, and mentions others as well. The author's main additions to what is covered in the published literature authored by others, are his comparisons, which seem to have a general bias toward favouring the SEI work and its CMM.

## 3.5. Business benefits of SPI

The author made this part the shortest of the five parts. In it, the author indicates that an integration of improvements in software processes affecting organizational processes, management processes, engineering processes, and support processes is needed to make possible SPI in customer and suppliers processes for improvements in the organization's 'bottom line'. To support this view, the author summarizes four reports of experiences in Europe and three reports of experiences in the USA. From these, the author points to six lessons learned:

- align SPI with business needs and customer satisfaction;
- focus investments in SPI to get significant benefits;
- demonstrate and communicate about the benefits of SPI;
- be patient for SPI's benefits to become manifest;
- use an incremental approach in applying SPI; and
- sell the benefits of SPI to the whole organization.

 

Software process improvement is a long-term activity, with most of its qualitative and quantitative gains coming from the non-technical side. Most of the gains come from SPI's focus on changing the on-the-job behaviour of people in doing and managing software work.

## 4. CONCLUSION

The real focus of this book is on discipline in software work processes, something that appears to be applicable in software maintenance processes.

NED CHAPIN
InfoSci Inc., Box 7117
Menlo Park CA 94026–7117, USA
Email: NedChapin@acm.org

## References

Chapin, N. (1988) 'Software maintenance life cycle', *Proceedings Conference on Software Maintenance—1988*, IEEE Computer Society, Los Alamitos CA, pp. 6–13.

Masengale, D. and Schrank, M. (1997) 'Model-based assessments applied to software maintenance processes', *Journal of Software Maintenance: Research and Practice*, **9**(2), 85–101.

 